Proof Internalization in Generalized Frege Systems for Classical Logic

Yury Savateev*
University of Bern

Abstract

We present a general method for inserting proofs in Frege systems for classical logic that produces systems that can internalize their own proofs.

Introduction.

The Logic of Proofs was introduced by Artemov in [1] (a survey article can be found at [2]). It is based on a notion of proof polynomials, a sort of templates for proofs, which allows to talk about proofs and propositions in the same language. In a sense it is a refinement of modal logic — a formula $\Box A$, which is typically interpreted as *A is provable* or *A is known* is replaced by an expression of the form $t : A$, that can be read as *t is a formal proof of A* or *A is known because of the evidence provided by t*. This approach helps to avoid the problem of logical omnisciece — when an agent knows all logical consequences of his assumptions (see [3]) and gives good ways to explore the semantics of provability. The central problem in justification logics is the realization theorem — how to replace all the boxes in a modal formula with corresponding justifications. An uniform way of doing it for different justification logics was presented in [6].

The structure of proof polynomials in justification logic is determined by the Hilbert-style system for classical logic. In this paper we offer a generalized approach to achieving the goals of justification logic that works on different styles of proof systems. We present a purely syntactical way of putting proofs themselves (not templates buit from external constants) inside a general class of what we call generalized Frege systems. This provides a uniform way to turn any system in this class into a system closed under the notion of proof internalization — any proof can be put inside a derivable object of the system and reasoned about using already existing rules. Our approach is different from the usual justification logics — instead of realizing modal logics we put proofs of a system for classical logic inside it. The main idea is that doing so we naturally arrive to a system that corresponds to a modal logic without adding anything else. The generalized Frege systems are a pretty wide class, and we only require them to hold some basic properties with respect to the classical logic in order for the method to work.

*The research was partly supported by Hasler Foundation.

While working with systems defined in such general way it is hard to provide a uniform realization procedure. Nevertheless, we were able to devise a uniform method of realization for some of the most commonly used systems for classical logic, like Hilbert-style systems and sequent systems with certain properties. Previous attempts of building justification system for sequent-based proofs (see [4]) were not entirely successful — the proof terms did not have enough information to recover the proof and it used external labels in proof terms.

Generalized Frege Systems

In this section we will define what we will mean by generalized Frege system $\mathcal{F}$.

The Language of the System $\mathcal{F}$

First we assume that we have a countable set of *propositional variables* $Prop = \{x_0, x_1, \ldots\}$. Then we have connectives to make formulas out of them and meta-connectives to build the derivable objects or DOs of our system (sequents).

We assume that the structure of formulas and DOs is given by a context-free grammar with the following restrictions:

1. There is a non-terminal symbol P that can be replaced by any propositional variable, and those rules are the only ones that mention propositional variables. This allows us to prohibit references to specific propositional variables in the axioms and rules descriptions.

2. There is a non-terminal symbol F rules for which can contain only terminal symbols for formula connectives and itself plus the additional rule $F \to P$.

3. There is a non-terminal symbol S rules for which can contain only terminal symbols for meta-connectives and non-terminals.

The set of all formulas (strings derivable from F) of our language we denote by $\mathrm{Fm}_{\mathcal{F}}$. We denote the set of all DOs (strings derivable from S) of our system by $\mathrm{DO}_{\mathcal{F}}$.

Axioms and Rules of the System $\mathcal{F}$

Axiom schemes are strings in the language of our grammar that do not have propositional variables in them (they can have the symbol P) and that can have labels on non-terminal symbols.

Axiom instances are a subset of $\mathrm{DO}_{\mathcal{F}}$. Each axiom instance can be derived from an axiom scheme with the restriction that substrings derived from the same non-terminals with the same label are the same.

Rule scheme is a tuple of several such strings with shared labels. A rule can have parameters derivable from specified strings of our grammar and must have one or more premises and one result. An exapmle of a rule with parameters is weakening in sequent systems — the formula being added to the sequent is a parameter. A rule instance is defined as a tuple of the elements derivable from corresponding strings with the same restrictions as the axioms. Premises and the result in the rule instance must be members of $\mathrm{DO}_{\mathcal{F}}$.

Each rule scheme should have a name, so given a name and corresponding number of parameters and DOs, we should be able to produce a unique result of applying this rule to these premises, or to tell that the rule is unapplicable (this means that different schemes can have the same name, as long as it does not cause confusion). This allows us to view rules with fixed parameters as functions of different arities on $\mathrm{DO}_{\mathcal{F}}$. We can also define other functions on $\mathrm{DO}_{\mathcal{F}}$ using rule schemes.

In all the examples that are given in this paper all rules have at most 2 premises. So we will denote the set of all unary rule names by $URule$ and the set of all binary rule names by $BRule$ and assume that those two sets contain all the rules of our system.

The structure of the system guarantees that it respects elementary substitution — we can replace one propositional variable in a DO with another and an axiom remains an axiom and a rule remains applicable and the result of it is predictable.

Proofs in the System $\mathcal{F}$

A generalized proof is a tree with DOs on the leaves and rule names (with parameters, if need be) on the internal nodes. Arity of rules should correspond to the number of sons of the node and it should be clear which son corresponds to which premise.

Each proof should have a result — the DO that this proof proves. If the tree has only one node we define the result to be the DO on it. The result of a non-trivial proof is the result of applying the rule, whose name is written on the root to the results of corresponding subproofs. If the rule is not applicable it is convenient to define the result to be the same as that of the subproof that corresponds to the last premise. The proof is called *clean* if it does not have unapplicable rules. Taking the result of a proof is a total function from the set of all proofs to $\mathrm{DO}_{\mathcal{F}}$.

A whole tree proof is a proof written in a more traditional way — as a tree with DOs and rule names on every node. A whole tree proof can be constructed from a generalized proof by putting results of corresponding subtrees on the nodes.

The DO A is said to be provable in the system $\mathcal{F}$ (denoted by $\vdash_{\mathcal{F}} A$) if it is a result of a proof with only axiom instances on its leaves. The DO A is said to be provable in the system $\mathcal{F}$ from a finite set of hypotheses $\{A_1, A_2, \ldots, A_n\}$ (denoted by $A_1, A_2, \ldots, A_n \vdash_{\mathcal{F}} A$) if it is a result of a proof where we allow members of $\{A_1, A_2, \ldots, A_n\}$ to be on leaves as well as axioms.

Refinement

A generalized Frege system $\mathcal{G}$ is said to be a refinement of generalized Frege system $\mathcal{F}$ if there is a surjective mapping $t : \mathrm{DO}_{\mathcal{G}} \to \mathrm{DO}_{\mathcal{F}}$ such that if $A_1, \ldots, A_n \vdash_{\mathcal{G}} A$ then $t(A_1), \ldots, t(A_n) \vdash_{\mathcal{F}} t(A)$; and $A_1, \ldots, A_n \vdash_{\mathcal{F}} A$ if and only if there exists $A'_1, \ldots, A'_n, A' \in \mathrm{DO}_{\mathcal{G}}$ such that $A'_1, \ldots, A'_n \vdash_{\mathcal{G}} A'$ and $t(A'_i) = A_i$.

Translation to Logic

We must define what it means for a generalized Frege system to represent a logic. In this paper we only consider classical logic and modal logic K (classical logic plus the axiom $\Box(F \to G) \to (\Box F \to \Box G)$ and a rule $\frac{F}{\Box F}$).

We should have a way to translate our DOs into formulas of the logic and formulas to our objects. Let Fm denote the set of formulas of the logic. We fix two mappings $s : \mathrm{Fm} \to \mathrm{DO}_{\mathcal{F}}$ and $f : \mathrm{DO}_{\mathcal{F}} \to \mathrm{Fm}$ that do the translations and demand that they respect elementary substitution and the following condition: for all $F \in \mathrm{Fm}$ the formula $F \leftrightarrow f(s(F))$ is a tautology.

Soundness means that all provable DOs of our system correspond to tautologies, and all unprovable DOs correspond to non-tautological formulas. So we say our system is sound if $\vdash_{\mathcal{F}} A$ if and only if $f(A)$ is a tautology.

Completeness means that only tautologies correspond to provable DOs. So we say our system is complete if the formula $F \in \mathrm{Fm}$ is a tautology if and only if $\vdash_{\mathcal{F}} s(F)$.

Deduction and Resolution

We say that the system $\mathcal{F}$ has the deduction property if for all DOs $A_1, \ldots, A_n$,A, and B, if $A_1, \ldots, A_n, A \vdash_{\mathcal{F}} B$, then $f(A_1), \ldots, f(A_n) \vdash f(A) \to f(B)$ in the logic.

The converse property, called the resolution property, is when if

$$f(A_1), \ldots, f(A_n) \vdash f(A) \to f(B)$$

in the logic, then $A_1, \ldots, A_n, A \vdash_{\mathcal{F}} B$.

Lemma 1. *If the system $\mathcal{F}$ has the resolution property, then for any formulas F and G of the logic, we have $s(F), s(F \to G) \vdash_{\mathcal{F}} s(G)$.*

From now on we will assume our system to be sound, complete, and have the deduction and resolution properties.

Disjunctive Function π

The disjunctive function π is a binary function on the DOs, and satisfy the following conditions for any DOs A, B, and C:

1. $A \vdash_{\mathcal{F}} \pi(A, B)$ and $B \vdash_{\mathcal{F}} \pi(A, B)$.
2. $\pi(A, B) \vdash_{\mathcal{F}} \pi(B, A)$.
3. If $A_1, \ldots, A_n \vdash_{\mathcal{F}} B$, then $\pi(A_1, C), \ldots, \pi(A_n, C) \vdash_{\mathcal{F}} \pi(B, C)$.
4. $\pi(A, A) \vdash_{\mathcal{F}} A$.

For classical logic such functions exist. For example we can take $s(f(A) \vee f(B))$.

Lemma 2. *In classical logic $f(\pi(A, B)) \leftrightarrow f(A) \vee f(B)$.*

Proof. Let us proove $f(\pi(A,B)) \to f(A) \vee f(B)$. Since $f(A) \to (f(A) \vee f(B))$ and $f(B) \to (f(A) \vee f(B))$, we have $A \vdash_{\mathcal{F}} s(f(A) \vee f(B))$ and $B \vdash_{\mathcal{F}} s(f(A) \vee f(B))$. Using properties of π we can show that $\pi(A,B) \vdash_{\mathcal{F}} s(f(A) \vee f(B))$. Therefore $f(\pi(A,B)) \to f(A) \vee f(B)$.

Now we proove $f(A) \vee f(B) \to f(\pi(A,B))$. Since $A \vdash_{\mathcal{F}} \pi(A,B)$ and $B \vdash_{\mathcal{F}} \pi(A,B)$, we can show that $f(A) \to f(\pi(A,B))$ and $f(B) \to f(\pi(A,B))$. Therefore $f(A) \vee f(B) \to f(\pi(A,B))$. □

Corollary 1. *For any classical formula A we have $\vdash_{\mathcal{F}} \pi(s(A), s(\neg A))$.*

For the sake of convenience we allow one argument of the function π to be empty and put $\pi(A,\varnothing) = \pi(\varnothing, A) = A$.

Usually there is a suitable function for π that can be defined by a simple rule scheme. So when we use π in the rule descriptions later in the article it is meant to be understood as a standart rule scheme that can be constructed from the the scheme for π.

Proof Internalization

Suppose our system $\mathcal{F}$ is sound and complete with respect to classical logic and has deduction and resolution properties. We want to expand it so it becomes closed under the notion of proof internalization — the proofs in the resulting system can be put inside the DOs and reasoned about.

We can describe this by extending the underlying grammar for the system $\mathcal{F}$ and adding new axioms and rules. It turns out that doing so we arrive at a system that corresponds to modal logic K in the same way that adding axioms about justification to classical logic corresponds to modal logic.

We will call this system $\mathcal{F}_{proof}$.

Language Expansion

We want to expand the language of $\mathcal{F}$ to accomodate proofs.

We impose a condition on the system $\mathcal{F}$ that we call *atomicity*: there is a scheme in the language of our grammar denoted by w that is derivable from S, has only one instance of the non-terminal P and no other non-terminals, and for all $x \in Prop$ we have $f(w_x) = x$ and $s(x) = w_x$, where w_x is a DO obtained by replacing P in w with x. Intuitively w_x is the simplest possible DO that corresponds to the simplest possible formula x.

We want to be able to describe in the language of our system all possible proofs of our system. That means that for every rule, we must have a higher order rule that represents adding the first rule to a proof inside the DO.

We add natural indices to the rule names of the system $\mathcal{F}$. The set of unary rules names for the system $\mathcal{F}_{proof}$ is

$$URule' = \{\alpha^n \mid \alpha \in URule, n \in \mathbb{N}\}$$

and the set of binary rules names is

$$BRule' = \{con^n \mid con \in BRule, n \in \mathbb{N}\}.$$

We introduce new terminal symbols for all the rules names. We introduce a new non-terminal Pr that will represent proofs. The rules for it are the

following: $Pr \to [S]$; $Pr \to Pr, \alpha$; $Pr \to Pr, Pr, con$; where $\alpha \in URule'$ and $con \in BRule'$. Also we add the rule $P \to (Pr)$.

Strings that can be derived from the symbol Pr naturally correspond to tree-like generalized proofs of our system. By $[A]$ we denote a trivial proof of object A — a proof without any steps, and then we build generalized proofs out of them with rule names.

Since the new grammar is an extension of the old one, results of application of old rules to new sequents are already defined.

Axioms and Rules

All axiom schemes of the system $\mathcal{F}$ remain in the system $\mathcal{F}_{proof}$. We also add the following axiom meta-scheme that represents the statement 'an axiom does not need proving': if A is an axiom of $\mathcal{F}_{proof}$, then so is $w_{([A])}$. This corresponds to axiom necessitation in justification logics.

The result of applying a rule α^0 where $\alpha \in URule$ is the same as the result of applying the rule α, the rule con^0 where $con \in BRule$ is the same as the rule con, and in other cases the rules work like the following:

$$\frac{\pi(w_{(Pr_1)}, S_1)}{\pi(w_{(Pr_1,\alpha^n),}, S_1)}\ \alpha^{n+1}, \qquad \frac{\pi(w_{(Pr_1)}, S_1) \quad \pi(w_{(Pr_2)}, S_2)}{\pi(w_{(Pr_1,Pr_2,con^n)}, \pi(S_1, S_2))}\ con^{n+1}.$$

Higher order rules work the same for all basic (with index 0) rules, so if we are to add new rules to our system we only need to specify how the basic rule work.

For realization purposes (so our system will be a refinement of modal logic) we need to add something else to our system. We add a binary *proof operator* $+$ (and a rule to the underlying grammar $Pr \to Pr, Pr, +$) and two unary rules of the system that deal with it:

$$\frac{\pi(w_{(Pr_2)}, S_1)}{\pi(w_{(Pr_1,Pr_2,+)}, S_1)}\ \pi^0_{l,Pr_1}, \qquad \frac{\pi(w_{(Pr_1)}, S_1)}{\pi(w_{(Pr_1,Pr_2,+)}, S_1)}\ \pi^0_{r,Pr_2}.$$

Notice that these rules have parameters of the type Pr. The operator $+$ serves the same purpose as $+$ in justification logic, although here we understand it differently.

Proof Results and the Flat Calculus

For any proof in the system $\mathcal{F}_{proof}$ there is a unique DO that corresponds to it. We will denote the result of a proof $\mathbb{A}$ by $res(\mathbb{A})$. Since our proofs are just graphical representation of trees, we can define the result of a proof as we did it before for trees with the addition that if $\mathbb{A}$ proves A and $\mathbb{B}$ proves B, then $\mathbb{A}, \mathbb{B}, +$ proves $\pi(A, B)$.

The mapping $\flat$ maps formulas and DOs to formulas and DOs that do not contain rule symbols and operators by recursively replacing all instances of $(\mathbb{A})$ with $([res(\mathbb{A})])$.

Consider the following system. The language is the same as in $\mathcal{F}_{proof}$ but without rule names and operators. It has the axioms of $\mathcal{F}_{proof}$ plus the following rules:

$$\frac{A}{\flat(res([A], \alpha))}\ \alpha, \qquad \frac{A \quad B}{\flat(res([A], [B], con))}\ con.$$

Here $\alpha \in URule'$ and $con \in BRule'$.

We will call this system $\mathcal{F}^{\flat}_{proof}$ or the flat calculus. For every object A provable in $\mathcal{F}^{\flat}_{proof}$ there is a clean proof of A.

Nested Structures

Nested sequent calculus was introduced in [5] and used in [6] for realization of different justification logics. Nested sequents generalize the notion of sequent and make it possible to create cut-free systems for modal logics. Nested structure we use here are used for the same purpose — we generalize our notion of DOs of our system and tie them to modal logic. Our nested structures do not provide a cut-free system for modal logic because they are built in a different way: we do not invent new special rules for dealing with modalities, but only use rules from the system $\mathcal{F}$. Also, unlike [6], our realization procedure does not depend on having a cut-free system.

Nested structures for the system $\mathcal{F}$ can be defined are just sequents in the language of the flat calculus $\mathcal{F}^{\flat}_{proof}$. In nested structure calculus we view boxes not as proofs, but simply as structural boxes that enrich our grammar similar to nested sequent calculus.

A *positive nested context* or PNC can be defined inductively in the following way. An empty context $\{\}$ is a positive nested context and if $\Sigma\{\}$ is a PNC, then so is $\pi(w_{([\Sigma\{\}])}, \Pi)$, where Π is a nested structure (or empty). The nested structure $\Sigma\{\Pi\}$ is obtained by replacing the hole $\{\}$ in the context $\Sigma\{\}$ with Π.

Now we can define *nested structure calculus* or NSC for the system $\mathcal{F}$. If A is an axiom instance of $\mathcal{F}$, then for any PNC $\Sigma\{\}$, $\Sigma\{A\}$ is an axiom of the NSC. For any $\alpha \in URule$ we have the rule

$$\frac{\Sigma\{A\}}{\Sigma\{res([A], \alpha)\}}.$$

For any $con \in BRule$ we have the rule

$$\frac{\Sigma\{A\} \quad \Sigma\{B\}}{\Sigma\{res([A], [B], con)\}}.$$

Rules from the original system $\mathcal{F}$ can not add any proof name and operators if we work in the language of $\mathcal{F}^{\flat}_{proof}$.

Ideologically PNCs play the similar role as contexts do in the nested sequent calculus — a way to apply rules deep inside the nested structure. We put more restrictions on the structure of PNCs then nested sequent calculus — the context in our nested structures is always in the 'leftmost' box inside the 'leftmost' box and so on, and it takes place of a whole DO and not a formula.

Our nested structures should correspond to modal formulas. So we must expand our definition of mappings f and s to accomodate structural boxes. We will call the expanded mappings f' and s'.

First we define the mapping f'. We take a nested structure Σ and for every structural box in it we introduce a fresh propositional variable. Then we replace the boxes with these variables, apply the function f to get a formula, and then we recursively substitute the variables $x_{[\Pi]}$, introduced for the structural box $[\Pi]$, with the formula $\Box(f'(\Pi))$.

We do the same thing for the mapping s'— we introduce fresh propositional variables for each subformula of the form $\Box G$, apply the mapping s, and recursively replace $x_{\Box G}$ with $[s'(G)]$. This means that for all formulas F, $s'(\Box F) = w_{[s'(F)]}$.

Theorem 1. *With mappings f' and s' the nested structure calculus is sound and complete with respect to modal logic K and has the resolution property.*

Proof. The fact that for any modal formula F, the formula $F \leftrightarrow f'(s'(F))$ is proven by induction on the modal depth of the formula.

Soundness follows directly from soundness of the system $\mathcal{F}$ and properties of the function π.

To prove completeness we need to prove that for any formulas F and G we can prove $s'(\Box(F \to G) \to (\Box F \to \Box G))$. Since

$$(\Box(F \to G) \to (\Box F \to \Box G) \leftrightarrow (\Box G \vee \neg\Box F \vee \neg\Box(F \to G))$$

it is sufficient to prove $\pi(s'(\Box G), \pi(s'(\neg\Box F), s'(\neg\Box(F \to G))))$. By Lemma 1 we know that $s(F), s(F \to G) \vdash_{\mathcal{F}} s(G)$. Therefore in nested structures $w_{[s'(F)]}, w_{[s'(F \to G)]} \vdash w_{[s'(G)]}$ which combined with properties of the function π gives us the desired result.

Resolution property also follows from the resolution property of the system $\mathcal{F}$.

□

Theorem 2. *A DO A is provable in $\mathcal{F}^{\flat}_{proof}$ if and only if A is provable in the NSC. The same is true with proving from hypotheses.*

Proof. Indeed the difference between nested structure calculus and $\mathcal{F}^{\flat}_{proof}$ is the same as the difference between sequent systems with shared contexts and without. Axioms of $\mathcal{F}^{\flat}_{proof}$ are axioms of the nested structure calculus, and axioms of NSC can be proven in $\mathcal{F}^{\flat}_{proof}$ usinng properties of the function π. The same goes with the rule application — applying a rule α in NSC to a context of structure depth n corresponds to applying rule α^n in $\mathcal{F}^{\flat}_{proof}$ and vice versa with additional proof elements needed whose existense is guaranteed by the properties of the function π.

□

Corollary 2. *The system $\mathcal{F}^{\flat}_{proof}$ is a sound and complete system for the modal logic K with the resolution property.*

Realization

Our goal is to prove that the system $\mathcal{F}_{proof}$ is a refinement of the nested structure calculus that correspond to modal logic.

In view of theorem 2 to prove realization it is sufficient to prove that $\mathcal{F}_{proof}$ is a refinement of $\mathcal{F}^{\flat}_{proof}$. As the mapping t that is required by the definition of refinement it is natural to take the mapping $\flat$.

Lemma 3. *If $\mathbb{A}$ is a clean proof of a DO A in the system $\mathcal{F}_{proof}$, then applying the operator $\flat$ to all DOs and rule parameters in it we will get a proof of the DO $\flat(A)$ in the system $\mathcal{F}^{\flat}_{proof}$.*

This lemma gives us the first requirement of the refinement definition. The other part is given by the following theorem.

Theorem (Realization Theorem)**.** *If* $A_1, \ldots, A_n \vdash_{\mathcal{F}^{\flat}_{proof}} A$*, then there are* $\mathcal{F}_{proof}$ *DOs* $A', A'_1, \ldots, A'_n$ *such that* $A'_1, \ldots, A'_n \vdash_{\mathcal{F}_{proof}} A$*;* $\flat(A') = A$*; and* $\flat(A'_i) = A_i$*.*

At the moment we do not have a general proof of the realization theorem. But we can prove it for some example systems.

Example 1: Hilbert-style System HC

In this section we translate the definition of a standart Hilbert-style system into our language and expand it to get a system that can internalize its own proofs. The resulting system is very similar to the justification logic JK.

Underlying Grammar

In this system our DOs are classical formulas themselves. We have formula connectives $\vee, \wedge, \rightarrow, \neg$ and brackets and no meta-connectives. Non-terminals are P, F, S. The rules of the grammar are the following:

$$\begin{array}{lcl} P & \rightarrow & x_i \\ F & \rightarrow & P \mid (F \vee F) \mid (F \wedge F) \mid (F \rightarrow F) \mid \neg F \\ S & \rightarrow & F \end{array}$$

Axioms and Rules

We take the standart axioms and rules for the Hilbert-style system for propositional classical logic and write them as schemes.

We have several axiom schemes:

1. $(F_1 \rightarrow (F_2 \rightarrow F_1))$.
2. $((F_1 \rightarrow (F_2 \rightarrow F_3)) \rightarrow ((F_1 \rightarrow F_2) \rightarrow (F_1 \rightarrow F_3)))$.
3. $((\neg F_1 \rightarrow \neg F_2) \rightarrow (F_2 \rightarrow F_1))$.
4. $(F_1 \rightarrow (F_2 \rightarrow (F_1 \wedge F_2)))$.
5. $((F_1 \wedge F_2) \rightarrow F_1)$.
6. $((F_1 \wedge F_2) \rightarrow F_2)$.
7. $(F_1 \rightarrow (F_1 \vee F_2))$.
8. $(F_1 \rightarrow (F_1 \vee F_2))$.
9. $((F_1 \rightarrow F_3) \rightarrow ((F_2 \rightarrow F_3) \rightarrow ((F_1 \vee F_2) \rightarrow F_3)))$.

and one rule scheme:

$$\frac{F_1 \quad (F_1 \rightarrow F_2)}{F_2} \times.$$

Here the set $URule$ is empty, and the set $BRule$ is $\{\times\}$.

Translation

Since we use the same language as classical logic we can just use the identity for both mappings s and f. The system is sound and complete. The deduction and resolution properties also hold.

Function π

For the disjunctive function π we can use disjunction itself using the scheme

$$\frac{F_1 \quad F_2}{(F_1 \vee F_2)}$$

Nested Structures

The scheme for w is just P.

The nested structures for HC are trivially translated to modal logic K by replacing structural boxes with $\Box$´s before the corresponding subformula.

Proof Internalization

The higher order versions of the only rule in the expanded version of the system HC that we call HC_{proof} look like this:

$$\frac{((Pr_1) \vee F_1) \quad ((Pr_2) \vee F_2)}{((Pr_1, Pr_2, \times^n) \vee (F_1 \vee F_2))} \times^{n+1}.$$

The rules for $+$ look like the following:

$$\frac{((Pr_2) \vee F_1)}{((Pr_1, Pr_2, +) \vee F_1)} \pi^0_{l,Pr_1}, \qquad \frac{((Pr_1) \vee F_1)}{((Pr_1, Pr_2, +) \vee F_1)} \pi^0_{r,Pr_2},$$

Example 2: Sequent System SC

In this section we take an example sequent system for classical logic and expand it to get a system that can internalize its own proofs — something that sequent systems for justification logic cannot do.

Underlying Grammar

We start with a context-free grammar that describes the structure of our sequents. We have formula connectives $\vee, \wedge, \rightarrow, \neg$ and brackets, meta-connectives are $\vdash, \bullet, \circ$ and comma. Non-terminals are $P, F, Fs, S, \Gamma, \Delta$. The rules of the grammar are the following:

$$\begin{array}{lcl} P & \rightarrow & x_i \\ F & \rightarrow & P \mid (F \vee F) \mid (F \wedge F) \mid (F \rightarrow F) \mid \neg F \\ Fs & \rightarrow & (F)^\circ \mid (F)^\bullet \\ \Gamma & \rightarrow & Fs\Delta \\ \Delta & \rightarrow & \varepsilon \mid , Fs\Delta \\ S & \rightarrow & \vdash \Gamma \end{array}$$

From S this grammar generates one-sided non-empty sequents with signed formulas.

Axioms and Rules

Axioms are given by the scheme $\vdash (F_1)^\circ, (F_1)^\bullet$.

The rules of the system are the following:

$$\frac{\vdash \Gamma_1}{\vdash Fs_1, \Gamma_1}\ \omega_{Fs_1}, \qquad \frac{\vdash Fs_1, Fs_1\Delta_1}{\vdash Fs_1\Delta_1}\ \gamma,$$

$$\frac{\vdash Fs_1, Fs_2\Delta_1}{\vdash Fs_2, Fs_1\Delta_1}\ \mu, \qquad \frac{\vdash Fs_1, \Gamma_1}{\vdash \Gamma_1, Fs_1}\ \nu,$$

$$\frac{\vdash (F_1)^\circ, (F_2)^\circ_1\Delta_1}{\vdash (F_1 \vee F_2)^\circ\Delta_1}\ \beta, \qquad \frac{\vdash (F_1)^\bullet\Delta_1 \quad \vdash (F_2)^\bullet\Delta_2}{\vdash (F_1 \vee F_2)^\bullet\Delta_1\Delta_2}\ \times,$$

$$\frac{\vdash (F_1)^\circ\Delta_1 \quad \vdash (F_2)^\circ\Delta_2}{\vdash (F_1 \wedge F_2)^\circ\Delta_1\Delta_2}\ \times, \qquad \frac{\vdash (F_1)^\bullet, (F_2)^\bullet\Delta_1}{\vdash (F_1 \wedge F_2)^\bullet\Delta_1}\ \beta,$$

$$\frac{\vdash (F_1)^\bullet, (F_2)^\circ\Delta_1}{\vdash (F_1 \to F_2)^\circ\Delta_1}\ \beta, \qquad \frac{\vdash (F_1)^\circ\Delta_1 \quad \vdash (F_2)^\bullet\Delta_2}{\vdash (F_1 \to F_2)^\bullet\Delta_1\Delta_2}\ \times,$$

$$\frac{\vdash (F_1)^\bullet\Delta_1}{\vdash (\neg F_1)^\circ\Delta_1}\ \chi, \qquad \frac{\vdash (F_1)^\circ\Delta_1}{\vdash (\neg F_1)^\bullet\Delta_1}\ \chi,$$

$$\frac{\vdash (F_1)^\circ, \Gamma_1 \quad \vdash (F_1)^\bullet, \Gamma_2}{\vdash \Gamma_1, \Gamma_2}\ \backslash. \qquad \frac{\vdash (F_1)^\circ \quad \vdash (F_1)^\bullet, \Gamma_2}{\vdash \Gamma_2}\ \backslash.$$

$$\frac{\vdash (F_1)^\circ, \Gamma_1 \quad \vdash (F_1)^\bullet}{\vdash \Gamma_1}\ \backslash.$$

So for this system $URule = \{\omega_{Fs}, \gamma, \mu, \nu, \beta, \chi\}$ and $BRule = \{\times, \backslash\}$. Notice that the rule ω has a parameter of type Fs and there are several schemes with the same names, but at most one is applicable at any situation. In fact, this system is chosen as an example because it was designed to have as few rule names as possible.

Translation

As a mapping f we use the mapping given by the following inductive definition restricted to sequents:

- $f(x_i^\circ) = x_i$,
- $f((\neg A)^\circ) = \neg f(A^\circ)$,
- $f((A \vee B)^\circ) = f(A^\circ) \vee f(B^\circ)$,
- $f((A \wedge B)^\circ) = f(A^\circ) \wedge f(B^\circ)$,
- $f((A \to B)^\circ) = f(A^\circ) \to f(B^\circ)$,
- $f(A^\bullet) = \neg f(A^\circ)$
- $f(\vdash A_1^*, \ldots, A_m^\star) = (f(A_1^*) \vee \ldots \vee (f(A_m^\star))$.

Here $*, \star \in \{\circ, \bullet\}$.

Mapping s is defined simply by $s(A) = \vdash (A)^\circ$.

It is easy to see that this system is sound and complete with respect to classical logic. Deduction and resolution properties are provided by the presence of the cut rule (the rule $\backslash$).

Function π

For the disjunctive function π we take the function given by the following rule scheme:

$$\frac{\vdash \Gamma_1 \quad \vdash \Gamma_2}{\vdash \Gamma_1, \Gamma_2}$$

It can be easily seen that it satisfies all of the requirements.

Nested Structures

The scheme for w is $\vdash (P)^\circ$.

The nested structures calculus for this system is similar to a signed version of the nested sequent calculus proposed by Brünnler in [5] with the difference that here we do not have boxes and diamonds as part of the language. But our PNC´s are a restricted versions of contexts in his system. Indeed our contexts can only be in the leftmost position and occupy a whole structural box. Also that system is cut-free but has a special rules to emulate the k axiom. We do it using the rule $\backslash$.

Proof Internalization

The higher order rules in the expanded version of the system SC that we call SC_{proof} look like this:

$$\frac{\vdash (Pr_1)^\circ \Delta_1}{\vdash (Pr_1, \alpha^n)^\circ \Delta_1}\ \alpha^{n+1}, \qquad \frac{\vdash (Pr_1)^\circ \Delta_1 \quad \vdash (Pr_2)^\circ \Delta_2}{\vdash (Pr_1, Pr_2, con^n)^\circ \Delta_1 \Delta_2}\ con^{n+1}.$$

Here $\alpha \in \{\omega_{Fs}, \gamma, \mu, \nu, \beta, \chi\}$ and $con \in \{\times, \backslash\}$.

The rules for $+$ look like the following:

$$\frac{\vdash (Pr_2)^\circ \Delta_1}{\vdash (Pr_1, Pr_2, +)^\circ \Delta_1}\ \pi^0_{l,Pr_1}, \qquad \frac{\vdash (Pr_1)^\circ \Delta_1}{\vdash (Pr_1, Pr_2, +)^\circ \Delta_1}\ \pi^0_{r,Pr_2},$$

Proof of the Realization Theorem

Despite the fact that we do not have the proof of the realization theorem in the most general case, the examples given show enough common structure to allow for a uniform proof of the realization theorem for them. This proof also applies to many other systems built this way, and may be possibly generalized even more.

Let us reiterate the realization theorem for our examples:

Theorem (Realization Theorem)**.** *If A is provable in $\mathrm{HC}^\flat_{proof}$ or $\mathrm{SC}^\flat_{proof}$ then there exists A' provable in HC_{proof} or SC_{proof} correspondingly, such that $\flat(A') = A$.*

Positive and Negative Instances

The thing our examples have in common is that they are both built using standartly understood connectives. This allows us to be able to naturally define whether a given instance of a subformula is positive or negative in a DO of a corresponding flat calculus. They also share another property that we call smooth context substitution that we will define and prove later. This allows for unified realization procedure. In what follows, by $\mathcal{F}$ we will understand one of the systems SC or HC.

Annotated DOs and Proofs

A proof subformula of a DO is a subformula that is a proof. An *annotated derivable object* or ADO is a $\mathcal{F}^{\flat}_{proof}$ DO whose proof subformulas (which are just trivial proofs) are labelled with $\mathcal{F}_{proof}$ proofs.

A *realization* of an ADO (or of a subformula, or a rule parameter) is a $\mathcal{F}_{proof}$ DO (or a subformula, or a rule parameter) that is obtained by substituting all proofs at the top level with corresponding annotations.

We will say that annotation of a DO is *coordinated*, if for every proof subformula $[S]$ annotated with a proof $\mathbb{A}$, $\mathbb{A}$ is clean and $res(\mathbb{A})$ is a realization of the ADO S. Labeling each proof subformula with itself, for example, provides a coordinated annotation. A coordinated annotation of a sequent is defined by its realization of the sequent.

An *annotated proof* of a DO is a clean whole tree proof where all DOs and rule parameters in the proof are annotated.

Now we will define coordinated annotated proofs.

1. Annotations of all DOs in the proof must be coordinated.
2. Realization of an axiom instance must be an axiom instance in $\mathcal{F}_{proof}$.
3. Realization of a rule instance must be a valid rule instance in $\mathcal{F}_{proof}$.

Lemma 4. *If there is a coordinated annotated proof that proves a DO A from hypotheses $A_1, \ldots, A_n$; then there are $\mathcal{F}_{proof}$ DOs $A', A'_1, \ldots, A'_n$ such that $A'_1, \ldots, A'_n \vdash_{\mathcal{F}_{proof}} A$; $\flat(A') = A$; and $\flat(A'_i) = A_i$.*

Proof. Replacing all the DOs and parameters in the coordinated annotated proof by their realizations, we get a valid proof in the system $\mathcal{F}_{proof}$ that outputs the desired sequent. □

Now we need a uniform way of building coordinated annotated proofs for provable DOs of $\mathcal{F}^{\flat}_{proof}$.

Affection Relation

Consider all the instances of proof subformulas in the whole tree proof. On this set we impose a relation called *affection*. We will talk how instances *affect* each other. The relation is defined by the following:

- Negative proof subformulas of the final DO are not affected by anything.

- Positive subformulas of the axioms that do not have corresponding negative instances and positive instances in the hypotheses are not affected by anything.
- New positive subformulas, created by a rule application (like weakening) are not affected by anything.
- Positive subformulas of the axioms, that have corresponding negative instances are affected by those instances.
- If an instance of a positive subformula is not changed by given step of the proof this instance in the premise of the rule affects this instance in the conclusion. For example, when a contraction-like rule is used, a positive instance of a subformula in the contracted formula is affected by the two instances from the premise.
- If we apply a rule that changes a positive instance then this instance affects the resulting positive instance. Similarly, if the new instance is combined from two different instances, it is affected by both of them.
- Negative instances in the conlusion of a rule affect the same instances in the premises. Again, if it is a contraction, one instance affects two.
- In the premises of a cut-like rule the positive instances in one copy of the the cut formula affect corresponding negative instances in the other copy.

Transitive affection relation is the transitive closure of the affection relation.

Lemma 5. *An instance of a formula cannot transitively affect its proper subformulas.*

We can think about a directed graph whose vertices are instances of proof subformulas in the whole tree proof, and whose edges connect vertices that affect one another. If that graph is acyclic we will call the proof acyclic.

Lemma 6. *If $A_1, \ldots, A_n \vdash_{\mathcal{F}^{\flat}_{proof}} A$ then there is an acyclic proof of A from hypotheses $A_1, \ldots, A_n$ that does not contain rules π^n_{l,Pr_1} and π^n_{r,Pr_2},.*

This lemma follows directly from the similar statement for propositional variable instances in the system $\mathcal{F}$.

Smooth Schemes and Proofs

A *proof scheme* is a tree of strings in underlying grammar for $\mathrm{HC}^{\flat}_{proof}$ that they only use connectives and non-terminal F with shared labels, such that if we substitute any formulas into it (instances with same label get substituted by the same formula), then we get a valid whole tree proof in $\mathrm{HC}^{\flat}_{proof}$.

A *smooth scheme* is a string in underlying grammar, that uses only connectives and nonterminal F, of the same format as axiom schemes but with additional requirements: each label must be present at most twice, and if it is there twice one must be in a positive position and one in a negative one; and there must be such a proof scheme, that the smooth scheme is at the root of it, and transitive affection relation (defined naturally on the proof scheme) is acyclic and if we restrict it to the root it only connects negative instance with

a positive instance with the same label. For example: $(F_1 \to F_1)$ is a smooth scheme because there is the following proof scheme:

$$\dfrac{(F_1 \to (F_2 \to F_1) \to F_1)) \quad \dfrac{((F_1 \to ((F_2 \to F_1) \to F_1)) \to ((F_1 \to (F_2 \to F_1)) \to (F_1 \to F_1)))}{((F_1 \to (F_2 \to F_1)) \to (F_1 \to F_1))}}{(F_1 \to F_1)}$$

Similarly we can define a smooth rule scheme.

Lemma 7. *The following schemes are smooth:* $(F_1 \vee F_2) \to (F_2 \vee F_1)$*;* $(F_1 \wedge F_2) \to (F_2 \wedge F_1)$*;* $(F_1 \vee F_2) \leftrightarrow (\neg F_1 \to F_2)$*;* $(F_1 \wedge F_2) \leftrightarrow \neg(F_1 \to \neg F_2)$*;* $F_1 \leftrightarrow \neg\neg F_1$*.*

Smooth Contextual Substitution

A generalized formula context is a DO of $\mathcal{F}^{\flat}_{proof}$ where exactly one instance of a propositional variable is replaced by a hole $\{\}$. In systems $\mathrm{HC}^{\flat}_{proof}$ and $\mathrm{SC}^{\flat}_{proof}$ any formula may be substituted in place of a hole and it will give us a valid DO. Contexts are naturally positive or negative.

Theorem 3 (Smooth contextual substituion)**.** *If* $w_{[A]} \vdash_{\mathcal{F}^{\flat}_{proof}} w_{[A']}$*, then for every positive context* Γ *there is a proof of* $\Gamma\{[A]\} \vdash \Gamma\{[A']\}$ *such that each positive instance in the context (outside A) in the premise transitively affects exactly one positive instance in the conclusion — the one standing in the same place. The same is true for negative instances in the conclusion — they transitively affect negative instances in the premise. For negative contexts there is a similar proof of* $\Gamma\{[A']\} \vdash \Gamma\{[A]\}$*.*

Proof. First we prove this theorem for $\mathrm{HC}^{\flat}_{proof}$.

We do it by induction on formula/DO construction.

The base case is trivial — there are no variable instances in the context.

Suppose that for the positive context $\Gamma\{\}$ the theorem holds.

Consider the positive context $(B \to \Gamma\{\})$. We take the whole tree smooth substitution proof for the context $\Gamma\{\}$ that exists by the induction hypothesis and replace all the axiom instances F with

$$\dfrac{F \quad (F \to (B \to F))}{(B \to F)}$$

And all rule instances of the form $\frac{F \quad (F \to G)}{G}$ are replaced by

$$\dfrac{(B \to (F \to G)) \quad \dfrac{B \to F \quad ((B \to F) \to ((B \to (F \to G)) \to (B \to G)))}{((B \to (F \to G)) \to (B \to G))}}{(B \to G)}$$

Deeper rule instances (of the rule $\times^n$ with $n > 0$) are replaced by similar construction of the appropriate depth. The resulting proof will provide smooth substitution for the context $(B \to \Gamma\{\})$.

Consider the negative context $\neg\Gamma\{\}$. By a previous case there is a smooth substitution for the context $(\Gamma\{[A]\} \to \Gamma\{\})$. The smooth proof for $\neg\Gamma\{[A']\} \vdash$

$\neg\Gamma\{[A]\}$ will be the following proof adjusted for appropriate depth:

$$\dfrac{\neg\Gamma\{[A']\} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma\{[A]\} \to \Gamma\{[A]\}}}{\vdots}}{\Gamma\{[A]\} \to \Gamma\{[A']\}} \quad (\Gamma\{[A]\} \to \Gamma\{[A']\}) \to (\neg\Gamma\{[A']\} \to \neg\Gamma\{[A]\})}{\neg\Gamma\{[A']\} \to \neg\Gamma\{[A]\}}}{\neg\Gamma\{[A]\}}$$

For the contexts $(\Gamma\{\} \to B)$, $(\Gamma\{\} \vee B)$, and $(\Gamma\{\} \wedge B)$, we can use smooth schemes to turn them into $(\neg B \to \neg\Gamma\{\})$, $(\neg B \to \Gamma\{\})$, and $\neg(B \to \neg\Gamma\{\})$ respectively, apply constructions from previous cases and use smooth schemes to change them back to their form.

The only case left is the positive context $[\Gamma\{\}]$. We take the smooth substitution proof for the context $\Gamma\{\}$ that exists by the induction hypothesis and replace all the axiom instances F with $[F]$ and all the rule names $\times^n$ with $\times^{n+1}$. The resulting proof will provide smooth substitution for the context $[\Gamma\{\}]$.

The constructions for the negative context $\Gamma\{\}$ are similar.

The theorem statement for $\mathrm{SC}^{\flat}_{proof}$ follows from the fact that we can directly translate $\mathrm{HC}^{\flat}_{proof}$ proofs into $\mathrm{SC}^{\flat}_{proof}$ proofs using cut. □

Realization Procedure

Now we are ready to do realization. We will take acyclic $\mathcal{F}^{\flat}_{proof}$ proof and turn it into a coordinated annotated proof of the same DO.

The general outline is the following — we start with the labeling of the instances that are not affected by anything and do not have any own proof subformulas. We label them with themselves. Then we propagate the labeling to the instances affected by already labelled. Then we move on to instances that are not affected by anything, but have all their subformulas already labelled. We always need to have all instances that affect the current instance, and all its own proof subformulas to be already labelled, in order to label it. If the current instance is affected by a negative instance (it cannot be affected by more than one), then we label it with the same label. If a positive instance is affected by a single other positive instance, then it should be labelled either with the same label, or with the label that represent the rule used. Also a negative instance affected with a positive one (the cut was used) takes the same label. If a positive instance is built from two positive labelled instances (and therefore is affected by them) with a rule we label it according to the rule used. Because of acyclicity and Lemma 5 the process will terminate and we will get a coordinated annotated proof.

The only problem happens when a contraction occurs — an instance is combined from two instances that should have the same labels, but they may not. In this case we need to modify the proof in such a way that it does not change what is already labelled, but allows to realize these instances.

So we have two labelled instances $[A]_{\mathbb{A}}$ and $[A]_{\mathbb{B}}$ that have all the same labels on everything inside them, but the labels for them are possibly different. They are affecting a single other yet unlabelled positive instance.

In case of $\mathrm{SC}^{\flat}_{proof}$ there is only one rule scheme where this can happen — the rule scheme γ which demands the labels to be the same after the realization

in order to be a valid rule instance. Here the two instances $[A]_{\mathbb{A}}$ and $[A]_{\mathbb{B}}$ are positive. Consider the following coordinated annotated proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash ([\vdash A_1^{*_1}, \ldots, A_n^{*_n}]_{\mathbb{A}})^\circ}{\vdash ([\vdash A_1^{*_1}, \ldots, A_n^{*_n}, A_1^{*_1}, \ldots, A_n^{*_n}]_{(\mathbb{A},\mathbb{B},+)})^\circ}\,\pi^0_{r,[A_1^{*_1},\ldots,A_n^{*_n}]_{\mathbb{B}}}}{\cdots}}{\vdash ([\vdash A_1^{*_1}, A_1^{*_1}, \ldots, A_n^{*_n}, A_n^{*_n}]_{(\mathbb{A},\mathbb{B},+,\ldots)})^\circ}\,\gamma^1}{\vdash ([\vdash A_1^{*_1}, \ldots, A_n^{*_n}, A_n^{*_n}]_{(\mathbb{A},\mathbb{B},+,\ldots,\gamma^0)})^\circ}\,\nu^1}{\vdash ([\vdash A_2^{*_n}, A_2^{*_n}, \ldots, A_n^{*_n}, A_n^{*_n}, A_1^{*_1}]_{(\mathbb{A},\mathbb{B},+,\ldots,\gamma^0,\nu^0)})^\circ}}{\cdots}}{\vdash ([\vdash A_1^{*_1}, \ldots, A_n^{*_n}]_{(\mathbb{A},\mathbb{B},+,\ldots,\gamma^0,\nu^0,\ldots)})^\circ}$$

There is a similar proof that goes from the instance $[A]_{\mathbb{B}}$ to the instance labelled with the same label $(\mathbb{A}, \mathbb{B}, +, \ldots, \gamma^0, \nu^0, \ldots)$ (it uses π_l instead of π_r). Now we can smooth substitute two instances labelled $(\mathbb{A}, \mathbb{B}, +, \ldots, \gamma^0, \nu^0, \ldots)$ for $[A]_{\mathbb{A}}$ and $[A]_{\mathbb{B}}$ and now the contraction is valid for all already labelled instances. Since the substituion is smooth it does not change transitive affection for already existing instances, and does not change already existing labels. It does make a proof bigger, but does not increase its proof depth (maximal number of nested square brackets) and we can automatically label instances inside that are affected by already labelled instances outside with the same labels.

For $\mathrm{HC}^\flat_{proof}$ the contraction happens in instances of axioms 2 and 4. Here the two instances $[A_1]$ and $[A_2]$ are negative and affect a single positive instance. Consider the following annotated proof:

$$\dfrac{\dfrac{[A]_{(\mathbb{A})}}{[A \vee A]_{(\mathbb{A},\mathbb{B},+)}}\,\pi^0_{r,[A]_{\mathbb{B}}} \qquad \dfrac{[A \to A]_{(.)} \qquad \dfrac{[A \to A]_{(.)} \qquad [(A \to A) \to ((A \to A) \to ((A \vee A) \to A))]_{(.)}}{[(A \to A) \to ((A \vee A) \to A)]_{(.)}}\,\times^1}{[(A \vee A) \to A]_{(.)}}\,\times^1}{[A]_{(\mathbb{A},\mathbb{B},+,\ldots,\times^0)}}\,\times^1$$

There is a similar proof for the instance $[A]_{\mathbb{B}}$. Now we can take our axiom instance and replace it with a proof that takes instance of the same axiom, but with our instances labelled with the same thing $((\mathbb{A}, \mathbb{B}, +, \ldots, \times^0))$ and smooth substitute $[A]_{\mathbb{A}}$ and $[A]_{\mathbb{B}}$ for the negative ones. Again we made the contraction realizable, all contractions we introduced in the process are also realizable, and we did not change any existing labels or alter the transitive affection.

This completes the proof of th Realization Theorem for systems HC_{proof} and SC_{proof}.

About Cut Elimination

Despite the fact that the system $\mathrm{SC}^\flat_{proof}$ relies heavily on using cuts for realization, the only rule that violates subformula property in it is the classical cut $\backslash^0$. Since we do not have rules that modify negative instances of proofs, all negative instances come unchanged from the axioms. So we can apply standart cut elimination procedure and arrive at a cut-free proof in the traditional sense — no rule loses information.

Further Generalization

We can modify our construction to be able to realize more powerful modal logics then K.

The easiest one to do is adding positive introspection, the axiom 4: $\Box F \to \Box\Box F$. To build a system for this logic we need to add a new proof operator ! with new rule $Pr \to !(Pr)$ in the underlying grammar, new rule in the system:

$$\frac{\pi(w_{(Pr_1)}), S_1)}{\pi(w_{(!(Pr_1))}, S_1)}\ \iota^0,$$

and postulate that $res(!(\mathbb{A})) = w_{(\mathbb{A})}$. The corresponding rule in the nested structure calculus will be

$$\frac{\Gamma\{\pi(w_{[S_1]}, S_2)\}}{\Gamma\{\pi(w_{[w_{[S_1]}]}, S_2)\}}\ .$$

Also easy to add is negative introspection, the axiom 5: $\neg\Box F \to \Box\neg\Box F$. For that we will need a scheme, analogous to w, that will be its negative counterpart. We will denote it by $\bar{w}$ and require that for all $x \in Prop$ we have $f(\bar{w}_x) = \neg x$ and $s(\neg x) = \bar{w}_x$, where $\bar{w}_x$ is a DO obtained by replacing P in $\bar{w}$ with x. In HC and SC the schemes will be $\neg P$ and $\vdash (P)^\bullet$, respectively. We then add a new proof operator ? with new rule $Pr \to ?(Pr)$ in the underlying grammar, new rule in the system with the same name ι:

$$\frac{\pi(\bar{w}_{(Pr_1)}), S_1)}{\pi(w_{(?(Pr_1))}, S_1)}\ \iota^0,$$

and postulate that $res(?(\mathbb{A})) = \bar{w}_{(\mathbb{A})}$. The corresponding rule in the nested structure calculus will be

$$\frac{\Gamma\{\pi(\bar{w}_{[S_1]}, S_2)\}}{\Gamma\{\pi(w_{[\bar{w}_{[S_1]}]}, S_2)\}}\ .$$

The realization procedure still works with obvious amendments. By adding one or both of these constuctions to HC_{proof} or SC_{proof} we get systems that realize modal logics $K4$, $K5$, and $K45$.

Acknowledgements

I am very grateful to Prof. Artemov for his valuable insights and support.

References

[1] Sergei [N.] Artemov. Logic of proofs. *Annals of Pure and Applied Logic*, 67(1–3):29–59, May 1994.

[2] Sergei N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, March 2001.

[3] Artemov, S. N. and R. Kuznets, Logical omniscience as a computational complexity problem, in A. Heifetz, editor, *Proceedings of TARK 2009*, pp. 14–23, 2009.

[4] Vladimir [N.] Brezhnev. On the logic of proofs. In Kristina Striegnitz, editor, *Proceedings of the sixth ESSLLI Student Session, 13th European Summer School in Logic, Language and Information (ESSLLI'01)*, pages 35–46, Helsinki, August 2001. FoLLI.

[5] Kai Brünnler. Deep sequent systems for modal logic. *Archive for Mathematical Logic*, Volume 48(6):551–577, 2009.

[6] Kai Brünnler, Remo Goetschi, and Roman Kuznets. A syntactic realization theorem for justification logics. In Lev Beklemishev, Valentin Goranko, and Valentin Shehtman eds, *Advances in Modal Logic*, Volume 8, pages 39–58. College Publications, 2010.